Speed and Accuracy in Shallow and Deep Stochastic Parsing

Ronald M. Kaplan , Stefan Riezler , Tracy Holloway King
John T. Maxwell III, **Alexander Vasserman** and **Richard Crouch**
Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304
{kaplan|riezler|king|maxwell|avasserm|crouch}@parc.com

Abstract

This paper reports some experiments that compare the accuracy and performance of two stochastic parsing systems. The currently popular Collins parser is a shallow parser whose output contains more detailed semantically-relevant information than other such parsers. The XLE parser is a deep-parsing system that couples a Lexical Functional Grammar to a log-linear disambiguation component and provides much richer representations theory. We measured the accuracy of both systems against a gold standard of the PARC 700 dependency bank, and also measured their processing times. We found the deep-parsing system to be more accurate than the Collins parser with only a slight reduction in parsing speed.[1]

1 Introduction

In applications that are sensitive to the meanings expressed by natural language sentences, it has become common in recent years simply to incorporate publicly available statistical parsers. A state-of-the-art statistical parsing system that enjoys great popularity in research systems is the parser described in Collins (1999) (henceforth "the Collins parser"). This system not only is frequently used for off-line data preprocessing, but also is included as a black-box component for applications such as document summarization (Daume and Marcu, 2002), information extraction (Miller et al., 2000), machine translation (Yamada and Knight, 2001), and question answering (Harabagiu et al., 2001). This is because the Collins parser shares the property of robustness with other statistical parsers, but more than other such parsers, the categories of its parse-trees make grammatical distinctions that presumably are useful for meaning-sensitive applications. For example, the categories of the Model 3 Collins parser distinguish between heads, arguments, and adjuncts and they mark some long-distance dependency paths; these distinctions can guide application-specific postprocessors in extracting important semantic relations.

In contrast, state-of-the-art parsing systems based on deep grammars mark explicitly and in much more detail a wider variety of syntactic and semantic dependencies and should therefore provide even better support for meaning-sensitive applications. But common wisdom has it that parsing systems based on deep linguistic grammars are too difficult to produce, lack coverage and robustness, and also have poor run-time performance. The Collins parser is thought to be accurate and fast and thus to represent a reasonable trade-off between "good-enough" output, speed, and robustness.

This paper reports on some experiments that put this conventional wisdom to an empirical test. We investigated the accuracy of recovering semantically-relevant grammatical dependencies from the tree-structures produced by the Collins parser, comparing these dependencies to gold-standard dependencies which are available for a subset of 700 sentences randomly drawn from section 23 of the Wall Street Journal (see King et al. (2003)).

We compared the output of the XLE system, a deep-grammar-based parsing system using the English Lexical-Functional Grammar previously constructed as part of the Pargram project (Butt et al., 2002), to the same gold standard. This system incorporates sophisticated ambiguity-management technology so that all possible syntactic analyses of a sentence are computed in an efficient, packed representation (Maxwell and Kaplan, 1993). In accordance with LFG theory, the output

[1]This research has been funded in part by contract # MDA904-03-C-0404 awarded from the Advanced Research and Development Activity, Novel Intelligence from Massive Data program. We would like to thank Chris Culy whose original experiments inspired this research.

includes not only standard context-free phrase-structure trees but also attribute-value matrices (LFG's f(unctional) structures) that explicitly encode predicate-argument relations and other meaningful properties. XLE selects the most probable analysis from the potentially large candidate set by means of a stochastic disambiguation component based on a log-linear (a.k.a. maximum-entropy) probability model (Riezler et al., 2002). The stochastic component is also "ambiguity-enabled" in the sense that the computations for statistical estimation and selection of the most probable analyses are done efficiently by dynamic programming, avoiding the need to unpack the parse forests and enumerate individual analyses. The underlying parsing system also has built-in robustness mechanisms that allow it to parse strings that are outside the scope of the grammar as a shortest sequence of well-formed "fragments". Furthermore, performance parameters that bound parsing and disambiguation work can be tuned for efficient but accurate operation.

As part of our assessment, we also measured the parsing speed of the two systems, taking into account all stages of processing that each system requires to produce its output. For example, since the Collins parser depends on a prior part-of-speech tagger (Ratnaparkhi, 1996), we included the time for POS tagging in our Collins measurements. XLE incorporates a sophisticated finite-state morphology and dictionary lookup component, and its time is part of the measure of XLE performance.

Performance parameters of both the Collins parser and the XLE system were adjusted on a heldout set consisting of a random selection of $1/5$ of the PARC 700 dependency bank; experimental results were then based on the other 560 sentences. For Model 3 of the Collins parser, a beam size of 1000, and not the recommended beam size of 10000, was found to optimize parsing speed at little loss in accuracy. On the same heldout set, parameters of the stochastic disambiguation system and parameters for parsing performance were adjusted for a Core and a Complete version of the XLE system, differing in the size of the constraint-set of the underlying grammar.

For both XLE and the Collins parser we wrote conversion programs to transform the normal (tree or f-structure) output into the corresponding relations of the dependency bank. This conversion was relatively straightforward for LFG structures (King et al., 2003). However, a certain amount of skill and intuition was required to provide a fair conversion of the Collins trees: we did not want to penalize configurations in the Collins trees that encoded alternative but equally legitimate representations of the same linguistic properties (e.g. whether auxiliaries are encoded as main verbs or aspect features), but we also did not want to build into the conversion program transformations that compensate for information that Collins cannot provide without appealing to additional linguistic resources (such as identifying the subjects of infinitival complements). We did not include the time for dependency conversion in our measures of performance.

The experimental results show that stochastic parsing with the Core LFG grammar achieves a better F-score than the Collins parser at a roughly comparable parsing speed. The XLE system achieves 12% reduction in error rate over the Collins parser, that is 77.6% F-score for the XLE system versus 74.6% for the Collins parser, at a cost in parsing time of a factor of 1.49.

2 Stochastic Parsing with LFG

2.1 Parsing with Lexical-Functional Grammar

The grammar used for this experiment was developed in the ParGram project (Butt et al., 2002). It uses LFG as a formalism, producing c(onstituent)-structures (trees) and f(unctional)-structures (attribute value matrices) as output. The c-structures encode constituency and linear order. F-structures encode predicate-argument relations and other grammatical information, e.g., number, tense, statement type. The XLE parser was used to produce packed representations, specifying all possible grammar analyses of the input.

In our system, tokenization and morphological analysis are performed by finite-state transductions arranged in a compositional cascade. Both the tokenizer and the morphological analyzer can produce multiple outputs. For example, the tokenizer will optionaly lowercase sentence initial words, and the morphological analyzer will produce *walk +Verb +Pres +3sg* and *walk +Noun +Pl* for the input form *walks*. The resulting tokenized and morphologically analyzed strings are presented to the symbolic LFG grammar.

The grammar can parse input that has XML delimited named entity markup: *<company>Columbia Savings</company> is a major holder of so-called junk bonds*. To allow the grammar to parse this markup, the tokenizer includes an additional tokenization of the strings whereby the material between the XML markup is treated as a single token with a special morphological tag (+NamedEntity). As a fall back, the tokenization that the string would have received without that markup is also produced. The named entities have a single multiword predicate. This helps in parsing both because it means that no internal structure has to be built for the predicate and because predicates that would otherwise be unrecognized by the grammar can be parsed (e.g., *Cie. Financiere de Paribas*). As described in section 5, it was also important to use named entity markup in these experiments to more fairly match the analyses in the PARC 700 dependency bank.

To increase robustness, the standard grammar is augmented with a FRAGMENT grammar. This allows sentences to be parsed as well-formed chunks specified by the grammar, in particular as Ss, NPs, PPs, and VPs, with unparsable tokens possibly interspersed. These chunks have both c-structures and f-structures corresponding to them. The grammar has a fewest-chunk method for determining the correct parse.

The grammar incorporates a version of Optimality Theory that allows certain (sub)rules in the grammar to be prefered or disprefered based on OT marks triggered by the (sub)rule (Frank et al., 1998). The Complete version of the grammar uses all of the (sub)rules in a multi-pass system that depends on the ranking of the OT marks in the rules. For example, topicalization is disprefered, but the topicalization rule will be triggered if no other parse can be built. A one-line rewrite of the Complete grammar creates a Core version of the grammar that moves the majority of the OT marks into the NOGOOD space. This effectively removes the (sub)rules that they mark from the grammar. So, for example, in the Core grammar there is no topicalization rule, and sentences with topics will receive a FRAGMENT parse. This single-pass Core grammar is smaller than the Complete grammar and hence is faster.

The XLE parser also allows the user to adjust performance parameters bounding the amount of work that is done in parsing for efficient but accurate operation. XLE's ambiguity management technology takes advantage of the fact that relatively few f-structure constraints apply to constituents that are far apart in the c-structure, so that sentences are typically parsed in polynomial time even though LFG parsing is known to be an NP-complete problem. But the worst-case exponential behavior does begin to appear for some constructions in some sentences, and the computational effort is limited by a SKIMMING mode whose onset is controlled by a user-specified parameter. When skimming, XLE will stop processing the subtree of a constituent whenever the amount of work exceeds that user-specified limit. The subtree is discarded, and the parser will move on to another subtree. This guarantees that parsing will be finished within reasonable limits of time and memory but at a cost of possibly lower accuracy if it causes the best analysis of a constituent to be discarded. As a separate parameter, XLE also lets the user limit the length of medial constituents, i.e., constituents that do not appear at the beginning or the end of a sentence (ignoring punctuation). The rationale behind this heuristic is to limit the weight of constituents in the middle of the sentence but still to allow sentence-final heavy constituents. This discards constituents in a somewhat more principled way as it tries to capture the psycholinguistic tendency to avoid deep center-embedding. When limiting the length of medial constituents, cubic-time parsing is possible for sentences up to that length, even with a deep, non-context-free grammar, and linear parsing time is possible for sentences beyond that length.

The Complete grammar achieved 100% coverage of section 23 as unseen unlabeled data: 79% as full parses, 21% FRAGMENT and/or SKIMMED parses.

2.2 Dynamic Programming for Estimation and Stochastic Disambiguation

The stochastic disambiguation model we employ defines an exponential (a.k.a. log-linear or maximum-entropy) probability model over the parses of the LFG grammar. The advantage of this family of probability distributions is that it allows the user to encode arbitrary properties of the parse trees as feature-functions of the probability model, without the feature-functions needing to be independent and non-overlapping. The general form of conditional exponential models is as follows:

$$p_{\boldsymbol{\lambda}}(x|y) = Z_{\boldsymbol{\lambda}}(y)^{-1} e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)}$$

where $Z_{\boldsymbol{\lambda}}(y) = \sum_{x \in X(y)} e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)}$ is a normalizing constant over the set $X(y)$ of parses for sentence y, $\boldsymbol{\lambda}$ is a vector of log-parameters, $\boldsymbol{f}$ is a vector of feature-values, and $\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)$ is a vector dot product denoting the (log-)weight of parse x.

Dynamic-programming algorithms that allow the efficient estimation and searching of log-linear models from a packed parse representation without enumerating an exponential number of parses have been recently presented by Miyao and Tsujii (2002) and Geman and Johnson (2002). These algorithms can be readily applied to the packed and/or-forests of Maxwell and Kaplan (1993), provided that each conjunctive node is annotated with feature-values of the log-linear model. In the notation of Miyao and Tsujii (2002), such a *feature forest* Φ is defined as a tuple $\langle C, D, r, \gamma, \delta \rangle$ where C is a set of conjunctive nodes, D is a set of disjunctive nodes, $r \in C$ is the root node, $\gamma : D \to 2^C$ is a conjunctive daughter function, and $\delta : C \to 2^D$ is a disjunctive daughter function.

A dynamic-programming solution to the problem of finding most probable parses is to compute the weight ϕ_d of each disjunctive node as the maximum weight of its conjunctive daugher nodes, i.e.,

$$\phi_d = \max_{c \in \gamma(d)} \phi_c \tag{1}$$

and to recursively define the weight ϕ_c of a conjunctive node as the product of the weights of all its descendant disjunctive nodes and of its own weight:

$$\phi_c = \prod_{d \in \delta(c)} \phi_d \; e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(c)} \tag{2}$$

Keeping a trace of the maximally weighted choices in a computaton of the weight ϕ_r of the root conjunctive node

r allows us to efficiently recover the most probable parse of a sentence from the packed representation of its parses.

The same formulae can be employed for an efficient calculation of probabilistic expectations of feature-functions for the statistical estimation of the parameters $\boldsymbol{\lambda}$. Replacing the maximization in equation 1 by a summation defines the *inside weight* of disjunctive node. Correspondingly, equation 2 denotes the inside weight of a conjunctive node. The *outside weight* ψ_c of a conjunctive node is defined as the outside weight of its disjunctive mother node(s):

$$\psi_c = \sum_{\{d|c\in\gamma(d)\}} \psi_d \tag{3}$$

The outside weight of a disjunctive node is the sum of the product of the outside weight(s) of its conjunctive mother(s), the weight(s) of its mother(s), and the inside weight(s) of its disjunctive sister(s):

$$\psi_d = \sum_{\{c|d\in\delta(c)\}} \{\psi_c \, e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(c)} \prod_{\{d'|d'\in\delta(c),d'\neq d\}} \phi_{d'}\} \tag{4}$$

From these formulae, the conditional expectation of a feature-function f_i can be computed from a chart with root node r for a sentence y in the following way:

$$\sum_{x\in X(y)} \frac{e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)} f_i(x)}{Z_{\boldsymbol{\lambda}}(y)} = \sum_{c\in C} \frac{\phi_c \psi_c f_i(c)}{\phi_r} \tag{5}$$

Formula 5 is used in our system to compute expectations for discriminative Bayesian estimation from partially labeled data using a first-order conjugate-gradient routine. For a more detailed description of the optimization problem and the feature-functions we use for stochastic LFG parsing see Riezler et al. (2002). We also employed a combined ℓ_1 regularization and feature selection technique described in Riezler and Vasserman (2004) that considerably speeds up estimation and guarantees small feature sets for stochastic disambiguation. In the experiments reported in this paper, however, dynamic programming is crucial for efficient stochastic disambiguation, i.e. to efficiently find the most probable parse from a packed parse forest that is annotated with feature-values. There are two operations involved in stochastic disambiguation, namely calculating feature-values from a parse forest and calculating node weights from a feature forest. Clearly, the first one is more expensive, especially for the extraction of values for non-local feature-functions over large charts. To control the cost of this computation, our stochastic disambiguation system includes a user-specified parameter for bounding the amount of work that is done in calculating feature-values. When the user-specified threshold for feature-value calculation is reached, this computation is discontinued, and the dynamic programming calculation for most-probable-parse search is computed from the current feature-value annotation of the parse forest. Since feature-value computation proceeds incrementally over the feature forest, i.e. for each node that is visited all feature-functions that apply to it are evaluated, a complete feature annotation can be guaranteed for the part of the and/or-forest that is visited until discontinuation. As discussed below, these parameters were set on a held-out portion of the PARC700 which was also used to set the Collins parameters.

In the experiments reported in this paper, we used a threshold on feature-extraction that allowed us to cut off feature-extraction in 3% of the cases at no loss in accuracy. Overall, feature extraction and weight calculation accounted for 5% of the computation time in combined parsing and stochastic selection.

3 The Gold-Standard Dependency Bank

We used the PARC 700 Dependency Bank (DEPBANK) as the gold standard in our experiments. The DEPBANK consists of dependency annotations for 700 sentences that were randomly extracted from section 23 of the UPenn Wall Street Journal (WSJ) treebank. As described by (King et al., 2003), the annotations were boot-strapped by parsing the sentences with a LFG grammar and transforming the resulting f-structures to a collection of dependency triples in the DEPBANK format. To prepare a true gold standard of dependencies, the tentative set of dependencies produced by the robust parser was then corrected and extended by human validators[2]. In this format each triple specifies that a particular relation holds between a head and either another head or a feature value, for example, that the SUBJ relation holds between the heads *run* and *dog* in the sentence *The dog ran*. Average sentence length of sentences in DEPBANK is 19.8 words, and the average number of dependencies per sentence is 65.4. The corpus is freely available for research and evaluation, as are documentation and tools for displaying and pruning structures.[3]

In our experiments we used a Reduced version of the DEPBANK, including just the minimum set of dependencies necessary for reading out the central semantic relations and properties of a sentence. We tested against this Reduced gold standard to establish accuracy on a lower bound of the information that a meaning-sensitive application would require. The Reduced version contained all the argument and adjunct dependencies shown in Fig. 1, and a few selected semantically-relevant features, as shown in Fig. 2. The features in Fig. 2 were chosen be-

[2]The resulting test set is thus unseen to the grammar and stochastic disambiguation system used in our experiments. This is indicated by the fact that the upperbound of F-score for the best matching parses for the experiment grammar is in the range of 85%, not 100%.

[3]http://www2.parc.com/istl/groups/nltt/fsbank/

Function	Meaning
adjunct	adjuncts
aquant	adjectival quantifiers (many, etc.)
comp	complement clauses (that, whether)
conj	conjuncts in coordinate structures
focus_int	fronted element in interrogatives
mod	noun-noun modifiers
number	numbers modifying nouns
obj	objects
obj_theta	secondary objects
obl	oblique
obl_ag	demoted subject of a passive
obl_compar	comparative *than*/*as* clauses
poss	possessives (*John's book*)
pron_int	interrogative pronouns
pron_rel	relative pronouns
quant	quantifiers (all, etc.)
subj	subjects
topic_rel	fronted element in relative clauses
xcomp	non-finite complements verbal and small clauses

Figure 1: Grammatical functions in DEPBANK.

cause it was felt that they were fundamental to the meaning of the sentences, and in fact they are required by the semantic interpreter we have used in a knowledge-based application (Crouch et al., 2002).

Feature	Meaning
adegree	degree of adjectives and adverbs (positive, comparative, superlative)
coord_form	form of a coordinating conjunction (e.g., *and, or*)
det_form	form of a determiner (e.g., *the, a*)
num	number of nouns (sg, pl)
number_type	cardinals vs. ordinals
passive	passive verb (e.g., *It was eaten.*)
perf	perfective verb (e.g., *have eaten*)
precoord_form	*either, neither*
prog	progressive verb (e.g., *were eating*)
pron_form	form of a pronoun (he, she, etc.)
prt_form	particle in a particle verb (e.g., *They threw it out.*)
stmt_type	statement type (declarative, interrogative, etc.)
subord_form	subordinating conjunction (e.g. *that*)
tense	tense of the verb (past, present, etc.)

Figure 2: Selected features for Reduced DEPBANK
.

As a concrete example, the dependency list in Fig. 3 is the Reduced set corresponding to the following sentence:

> He reiterated his opposition to such funding, but expressed hope of a compromise.

An additional feature of the DEPBANK that is relevant to our comparisons is that dependency heads are represented by their standard citation forms (e.g. the verb *swam* in a sentence appears as *swim* in its dependencies). We believe that most applications will require a conversion to canonical citation forms so that semantic relations can be mapped into application-specific databases or ontologies. The predicates of LFG f-structures are already represented as citation forms; for a fair comparison we ran the leaves of the Collins tree through the same stemmer modules as part of the tree-to-dependency translation. We also note that proper names appear in the DEPBANK as single multi-word expressions without any internal structure. That is, there are no dependencies holding among the parts of people names (*A. Boyd Simpson*), company names (*Goldman, Sachs & Co*), and organization names (*Federal Reserve*). This multiword analysis was chosen because many applications do not require the internal structure of names, and the identification of named entities is now typically carried out by a separate non-syntactic pre-processing module. This was captured for the LFG parser by using named entity markup and for the Collins parser by creating complex word forms with a single POS tag (section 5).

```
conj(coord∼0, express∼3)
conj(coord∼0, reiterate∼1)
coord_form(coord∼0, but)
stmt_type(coord∼0, declarative)
obj(reiterate∼1, opposition∼6)
subj(reiterate∼1, pro∼7)
tense(reiterate∼1, past)
obj(express∼3, hope∼15)
subj(express∼3, pro∼7)
tense(express∼3, past)
adjunct(opposition∼6, to∼11)
num(opposition∼6, sg)
poss(opposition∼6, pro∼19)
num(pro∼7, sg)
pron_form(pro∼7, he)
obj(to∼11, funding∼13)
adjunct(funding∼13, such∼45)
num(funding∼13, sg)
adjunct(hope∼15, of∼46)
num(hope∼15, sg)
num(pro∼19, sg)
pron_form(pro∼19, he)
adegree(such∼45, positive)
obj(of∼46, compromise∼54)
det_form(compromise∼54, a)
num(compromise∼54, sg)
```

Figure 3: Reduced dependency relations for *He reiterated his opposition to such funding, but expressed hope of a compromise.*

4 Conversion to Dependency Bank Format

A conversion routine was required for each system to transform its output so that it could be compared to the DEPBANK dependencies. While it is relatively straightforward to convert LFG f-structures to the dependency bank format because the f-structure is effectively a dependency format, it is more difficult to transform the output trees of the Model 3 Collins parser in a way that fairly allocates both credits and penalties.

LFG Conversion We discarded the LFG tree structures and used a general rewriting system previously developed for machine translation to rewrite the relevant f-structure attributes as dependencies (see King et al. (2003)). The rewritings involved some deletions of irrelevant features, some systematic manipulations of the analyses, and some trivial respellings. The deletions involved features produced by the grammar but not included in the PARC 700 such as negative values of PASS, PERF, and PROG and the feature MEASURE used to mark measure phrases. The manipulations are more interesting and are necessary to map systematic differences between the analyses in the grammar and those in the dependency bank. For example, coordination is treated as a set by the LFG grammar but as a single COORD dependency with several CONJ relations in the dependency bank. Finally, the trivial rewritings were used to, for example, change STMT-TYPE `decl` in the grammar to STMT-TYPE `declarative` in the dependency bank. For the Reduced version of the PARC 700 substantially more features were deleted.

Collins Model 3 Conversion An abbreviated representation of the Collins tree for the example above is shown in Fig. 4. In this display we have eliminated the head lexical items that appear redundantly at all the nonterminals in a head chain, instead indicating by a single number which daughter is the head. Thus, S˜2 indicates that the head of the main clause is its second daughter, the VP, and its head is its first VP daughter. Indirectly, then, the lexical head of the S is the first verb *reiterated*.

```
(TOP∼1
(S∼2 (NP-A∼1 (NPB∼1 He/PRP))
     (VP∼1 (VP∼1 reiterated/VBD
                  (NP-A∼1 (NPB∼2 his/PRP$
                                 opposition/NN)
                          (PP∼1 to/TO
                                (NPB∼2 such/JJ
                                 funding/NN))))
            but/CC
            (VP∼1 expressed/VBD
                  (NP-A∼1 (NPB∼1 hope/NN)
                          (PP∼1 of/IN
                                (NP-A∼1 (NPB∼2 a/DT
                                 compromise/NN))))))))
```

Figure 4: Collins Model 3 tree for *He reiterated his opposition to such funding, but expressed hope of a compromise.*

The Model 3 output in this example includes standard phrase structure categories, indications of the heads, and the additional -A marker to distinguish arguments from adjuncts. The terminal nodes of this tree are inflected forms, and the first phase of our conversion replaces them with their citation forms (the verbs *reiterate* and *express*, and the decapitalized and standardized *he* for *He* and *his*). We also adjust for systematic differences in the choice of heads. The first conjunct tends to be marked as the head of a coordination in Model 3 output, whereas the dependency bank has a more symmetric representation: it introduces a new COORD head and connects that up to the conjunction, and it uses a separate CONJ relation for each of the coordinated items. Similarly, Model 3 identifies the syntactic markers *to* and *that* as the heads of complements, whereas the dependency bank treats these as selectional features and marks the main predicate of the complements as the head. These adjustments are carried out without penalty. We also compensate for the differences in the representation of auxiliaries: Model 3 treats these as main verbs with embedded complements instead of the PERF, PROG, and PASSIVE features of the DEPBANK, and our conversion flattens the trees so that the features can be read off.

The dependencies are read off after these and a few other adjustments are made. NPs under VPs are read off either as objects or adjuncts, depending on whether or not the NP is annotated with the argument indicator (-A) as in this example; the -A presumably would be missing in a sentence like *John arrived Friday*, and *Friday* would be treated as an ADJUNCT. Similarly, NP-As under S are read off as subject. In this example, however, this principle of conversion does not lead to a match with the dependency bank: in the DEPBANK grammatical relations that are factored out of conjoined structures are distributed back into those structures, to establish the correct semantic dependencies (in this case, that *he* is the subject of both *reiterate* and *express* and not of the introduced *coord*). We avoided the temptation of building coordinate distribution into the conversion routine because, first, it is not always obvious from the Model 3 output when distribution should take place, and second, that would be a first step towards building into the conversion routine the deep lexical and syntactic knowledge (essentially the functional component of our LFG grammar) that the shallow approach explicitly discounts[4].

For the same reasons our conversion routine does not identify the subjects of infinitival complements with particular arguments of matrix verbs. The Model 3 trees provide no indication of how this is to be done, and in many cases the proper assignment depends on lexical information about specific predicates (to capture, for example, the well-known contrast between *promise* and *persuade*).

Model 3 trees also provide information about certain

[4]However, we did explore a few of these additional transformations and found only marginal F-score increases.

long-distance dependencies, by marking with -g annotations the path between a filler and a gap and marking the gap by an explicit TRACE in the terminal string. The filler itself is not clearly identified, but our conversion treats all WH categories under SBAR as potential fillers and attempts to propagate them down the gap-chain to link them up to appropriate traces.

In sum, it is not a trivial matter to convert a Model 3 tree to an appropriate set of dependency relations, and the process requires a certain amount of intuition and skill. For our experiments we tried to define a conversion that gives appropriate credit to the dependencies that can be read from the trees without relying on an undue amount of sophisticated linguistic knowledge[5].

5 Experiments

We conducted our experiments by preparing versions of the test sentences in the form appropriate to each system. We used a configuration of the XLE parser that expects sentences conforming to ordinary text conventions to appear in a file separated by double line-feeds. A certain amount of effort was required to remove the part-of-speech tags and labeled brackets of the WSJ corpus in a way that restored the sentences to a standard English format (for example, to remove the space between *wo* and *n't* that remains when the POS tags are removed). Since the PARC 700 treats proper names as multiword expressions, we then augmented the input strings with XML markup of the named entities. These are parsed by the grammar as described in section 2. We used manual named entity markup for this experiment because our intent is to measure parsing technology independent of either the time or errors of an automatic named-entity extractor. However, in other experiments with an automatic finite-state extractor, we have found that the time for named-entity recognition is negligible (on the order of seconds across the entire corpus) and makes relatively few errors, so that the results reported here are good approximations of what might be expected in more realistic situations.

As input to the Collins parser, we used the part-of-speech tagged version of section 23 that was provided with the parser. From this we extracted the 700 sentences in the PARC 700. We then modified them to produce named entity input so that the parses would match the PARC 700. This was done by putting underscores between the parts of the named entity and changing the final part of speech tag to the appropriate one (usually NNP) if necessary. (The number of words indicated at the beginning of the input string was also reduced accordingly.) An example is shown in (1).

[5]The results of this conversion are available at http://www2.parc.com/istl/groups/nltt/fsbank/

(1) Sen. NNP Christopher NNP Dodd NNP $\longrightarrow$
Sen._Christopher_Dodd NNP

After parsing, the underscores were converted to spaces to match the PARC 700 predicates.

Before the final evaluation, $1/5$ of the PARC 700 dependency bank was randomly extracted as a heldout set. This set was used to adjust the performance parameters of the XLE system and the Collins parser so as to optimize parsing speed without losing accuracy. For example, the limit on the length of medial phrases was set to 20 words for the XLE system (see Sec. 2), and a regularizer penalty of 10 was found optimal for the ℓ_1 prior used in stochastic disambiguation. For the Collins parser, a beam size of 1000 was found to improve speed considerably at little cost in accuracy. Furthermore, the np-bracketing flag (npbflag) was set to 0 to produce an extended set of NP levels for improved argument/adjunct distinction[6]. The final evaluation was done on the remaining 560 examples. Timing results are reported in seconds of CPU time[7]. POS tagging of the input to the Collins parser took 6 seconds and this was added to the timing result of the Collins parser. Time spent for finite-state morphology and dictionary lookup for XLE is part of the measure of its timing performance. We did not include the time for dependency extraction or stemming the Collins output.

Table 1 shows timing and accuracy results for the Reduced dependency set. The parser settings compared are Model 3 of the Collins parser adjusted to beam size 1000, and the Core and Complete versions of the XLE system, differing in the size of the grammar's constraint-set. Clearly, both versions of the XLE system achieve a significant reduction in error rate over the Collins parser (12% for the core XLE system and 20% for the complete system) at an increase in parsing time of a factor of only 1.49 for the core XLE system. The complete version gives an overall improvement in F-score of 5% over the Collins parser at a cost of a factor of 5 in parsing time.

Table 1: Timing and accuracy results for Collins parser and Complete and Core versions of XLE system on Reduced version of PARC 700 dependency bank.

	time	prec.	rec.	F-score
LFG core	298.88	79.1	76.2	77.6
LFG complete	985.3	79.4	79.8	79.6
Collins 1000	199.6	78.3	71.2	74.6

[6]A beam size of 10000 as used in Collins (1999) improved the F-score on the heldout set only by .1% at an increase of parsing time by a factor of 3. Beam sizes lower than 1000 decreased the heldout F-score significantly.

[7]All experiments were run on one CPU of a dual processor AMD Opteron 244 with 1.8 GHz and 4GB main memory. Loading times are included in CPU times.

6 Conclusion

We presented some experiments that compare the accuracy and performance of two stochastic parsing systems, the shallow Collins parser and the deep-grammar-based XLE system. We measured the accuracy of both systems against a gold standard derived from the PARC 700 dependency bank, and also measured their processing times. Contrary to conventional wisdom, we found that the shallow system was not substantially faster than the deep parser operating on a core grammar, while the deep system was significantly more accurate. Furthermore, extending the grammar base of the deep system results in much better accuracy at a cost of a factor of 5 in speed.

Our experiment is comparable to recent work on reading off Propbank-style (Kingsbury and Palmer, 2002) predicate-argument relations from gold-standard treebank trees and automatic parses of the Collins parser. Gildea and Palmer (2002) report F-score results in the 55% range for argument and boundary recognition based on automatic parses. From this perspective, the nearly 75% F-score that is achieved for our deterministic rewriting of Collins' trees into dependencies is remarkable, even if the results are not directly comparable. Our scores and Gildea and Palmer's are both substantially lower than the 90% typically cited for evaluations based on labeled or unlabeled bracketing, suggesting that extracting semantically relevant dependencies is a more difficult, but we think more valuable, task.

References

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING2002, Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

D. Crouch, C. Condoravdi, R. Stolle, T.H. King, V. de Paiva, J. Everett, and D. Bobrow. 2002. Scalability of redundancy detection in focused document collections. In *Proceedings of Scalable Natural Language Understanding*, Heidelberg.

Hal Daume and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA.

Anette Frank, Tracy H. King, Jonas Kuhn, and John Maxwell. 1998. Optimality theory style constraint ranking in large-scale LFG grammars. In *Proceedings of the Third LFG Conference*.

Stuart Geman and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochatic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA.

Dan Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Asssociation for Computational Linguistics (ACL'01)*, Toulouse, France.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the Workshop on "Linguistically Interpreted Corpora" at the 10th Conference of the European Chapter of the Association for Computational Linguistics (LINC'03)*, Budapest, Hungary.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Spain.

John Maxwell and Ron Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, Seattle, WA.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference (HLT'02)*, San Diego, CA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP-1*.

Stefan Riezler and Alexander Vasserman. 2004. Gradient feature testing and ℓ_1 regularization for maximum entropy parsing. Submitted for publication.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Asssociation for Computational Linguistics (ACL'01)*, Toulouse, France.